计算机学院专业必修课

计算机组成

流水线处理器
形式建模综合方法

高小鹏

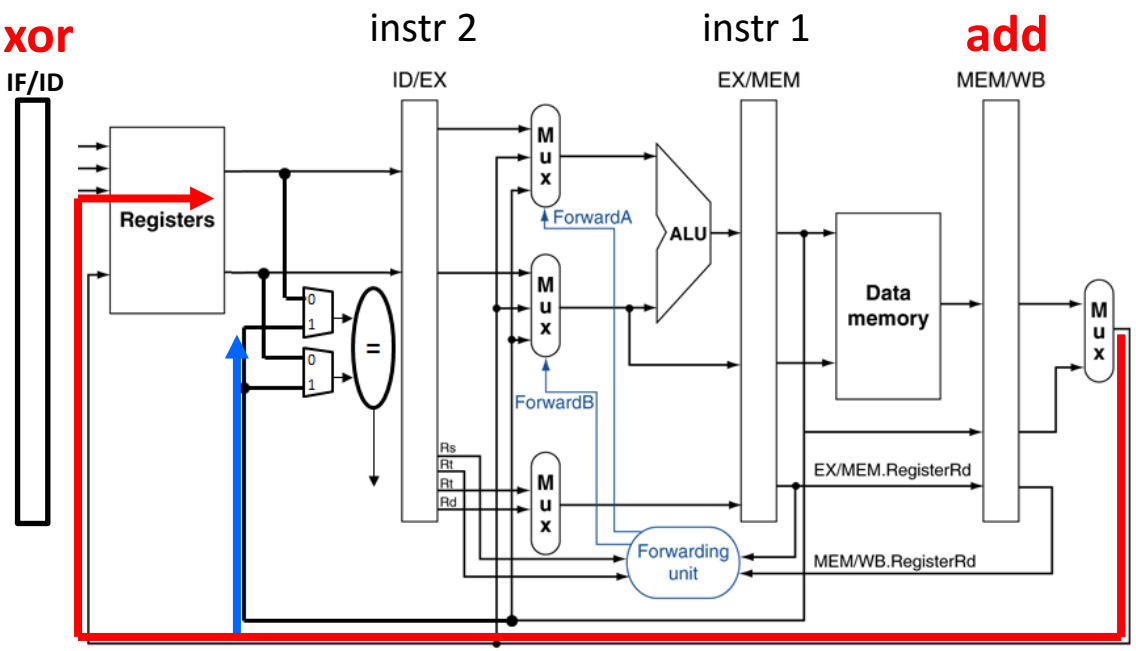
北京航空航天大学计算机学院

RF原设计在流水线中出错原因分析

- 场景：读R与写R同时发生
- 现象：软件执行错误
- 原因：无法利用MEM/WB的有效反馈
- 思
- 思路2：修改RF

读出错误数据

来不及了

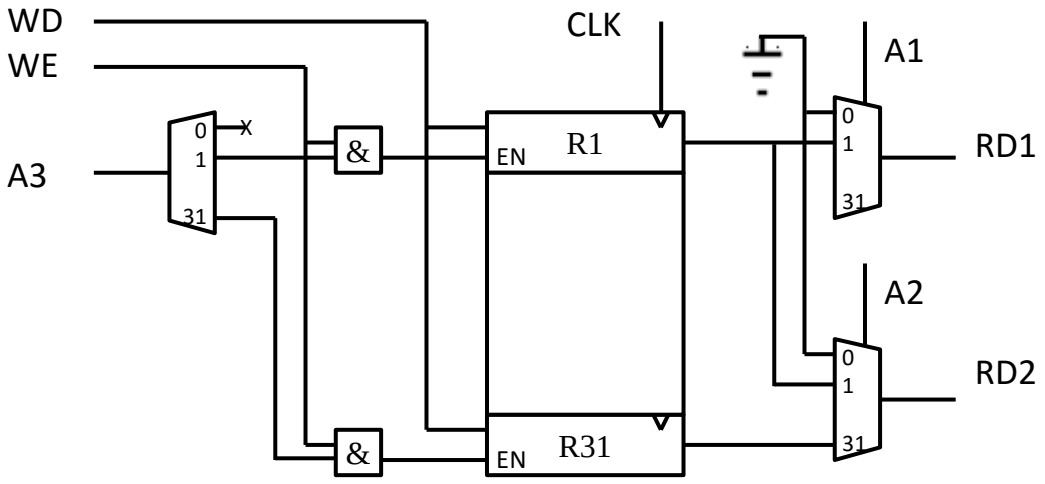


IF级	ID级	EX级	MEM级	WB级

地址	指令	CLK	PC	IF/ID	ID/EX	EX/MEM ⇨EX, ID	MEM/WB ⇨EX	RF
0	**add $1, $2, $3**	↑1	0→4	add				
4	**instr 1**	↑2	4→8	instr 1	add			
8	**instr 2**	↑3	0→4	instr 2	instr 1	add(结果)		
12	**xor $5, $1, $2**	↑4	0→4	xor(读R)	instr 2	instr 1	add(结果)	
16	**sub**	↑5	4→8	sub(读R)	xor(用R)	instr 2	instr 1	add(更新)

RF原设计在流水线中出错原因分析

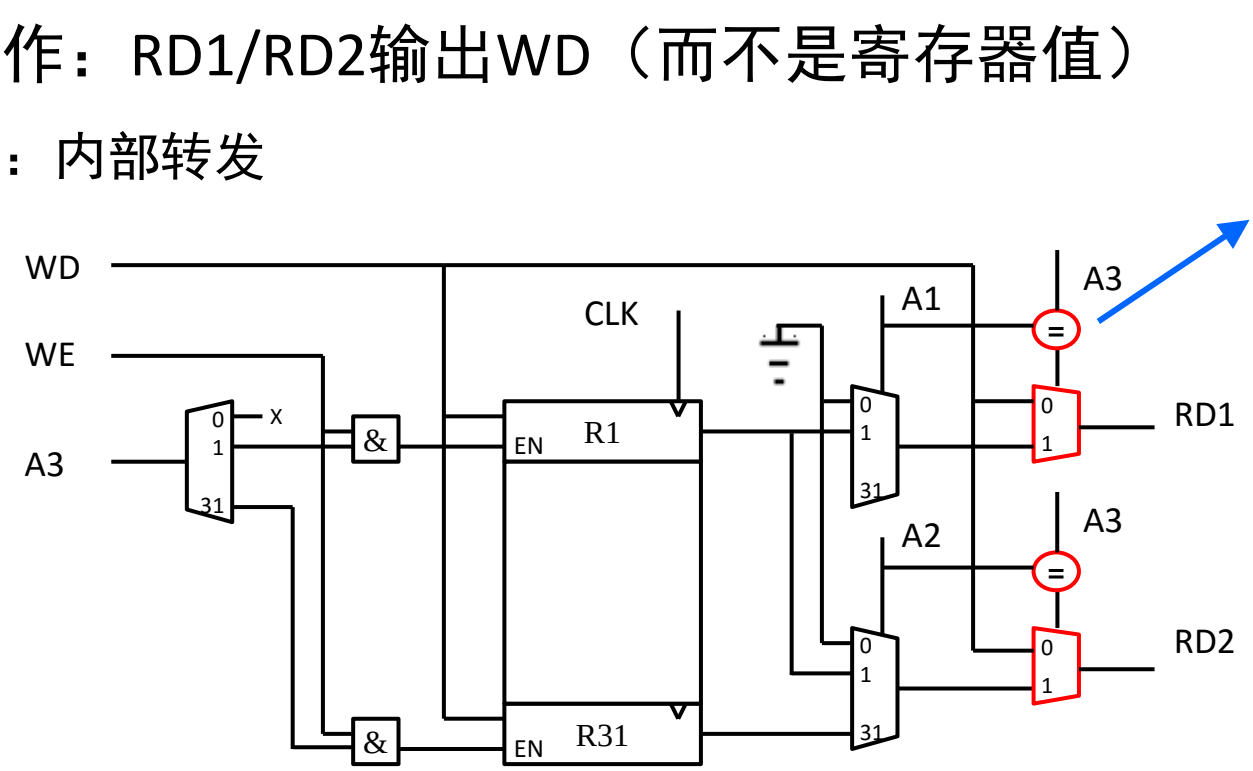
- 错误场景：写入寄存器与读出寄存器同时发生
  - RF：存储的是旧值
  - WD总线：新值
- RD1/RD2：输出的是旧值
  - 写入值滞后1个时钟周期



		IF级	ID级	EX级	MEM级	WB级		
地址	指令	CLK	PC	IF/ID	ID/EX	EX/MEM ⇨EX, ID	MEM/WB ⇨EX	RF
0	**add $1, $2, $3**	↑ 1	0→4	add				
4	**instr 1**	↑ 2	4→8	instr 1	add			
8	**instr 2**	↑ 3	0→4	instr 2	instr 1	add(结果)		
12	**xor $5, $1, $2**	↑ 4	0→4	xor(读R)	instr 2	instr 1	add(结果)	
16	**sub**	↑ 5	4→8	sub(读R)	xor(用R)	instr 2	instr 1	add(更新)

适用于流水线的RF内部结构设计

- 判断条件：写入寄存器与读出寄存器相同
  - A3 == A1 或者 A3 == A2
- 执行操作：RD1/RD2输出WD（而不是寄存器值）
  - 本质：内部转发



- Q：是否有遗漏条件？
- H：当前虽然不是写入操作，但是A3恰好等A1/A2！

开发流水线CPU的挑战

- 诉求：高性能；多类别；高效率；正确性
  - 高性能：全力转发（不允许出现任何可以避免的暂停）
  - 多类别：包含常见指令类别
  - 高效率：1~2周内完成开发工作
  - 正确性：能正确处理所有可能的冲突
- 挑战：如何保证正确性？
  - 高性能：导致结构复杂、设计规模大
  - 类别多：导致需要分析与处理的冲突增加
  - 高效率：导致开发与测试时间短

提纲

- 基础指令集
- 流水线基本架构
- 无转发数据通路构造方法
- 功能部件控制信号构造方法
- 数据冒险的一般性分析方法
- 暂停机制生成方法
- 转发机制生成方法
- 控制冒险处理机制

基础指令集与标准流水线

- 指令集
  - lw，sw，addu，subu，ori，lui，beq，j，jal，jalr
- 典型指令；可以支持大多数程序需求
- jal，jalr：涉及2个写入操作，PC写入，RF写入
  - 比较特殊的指令

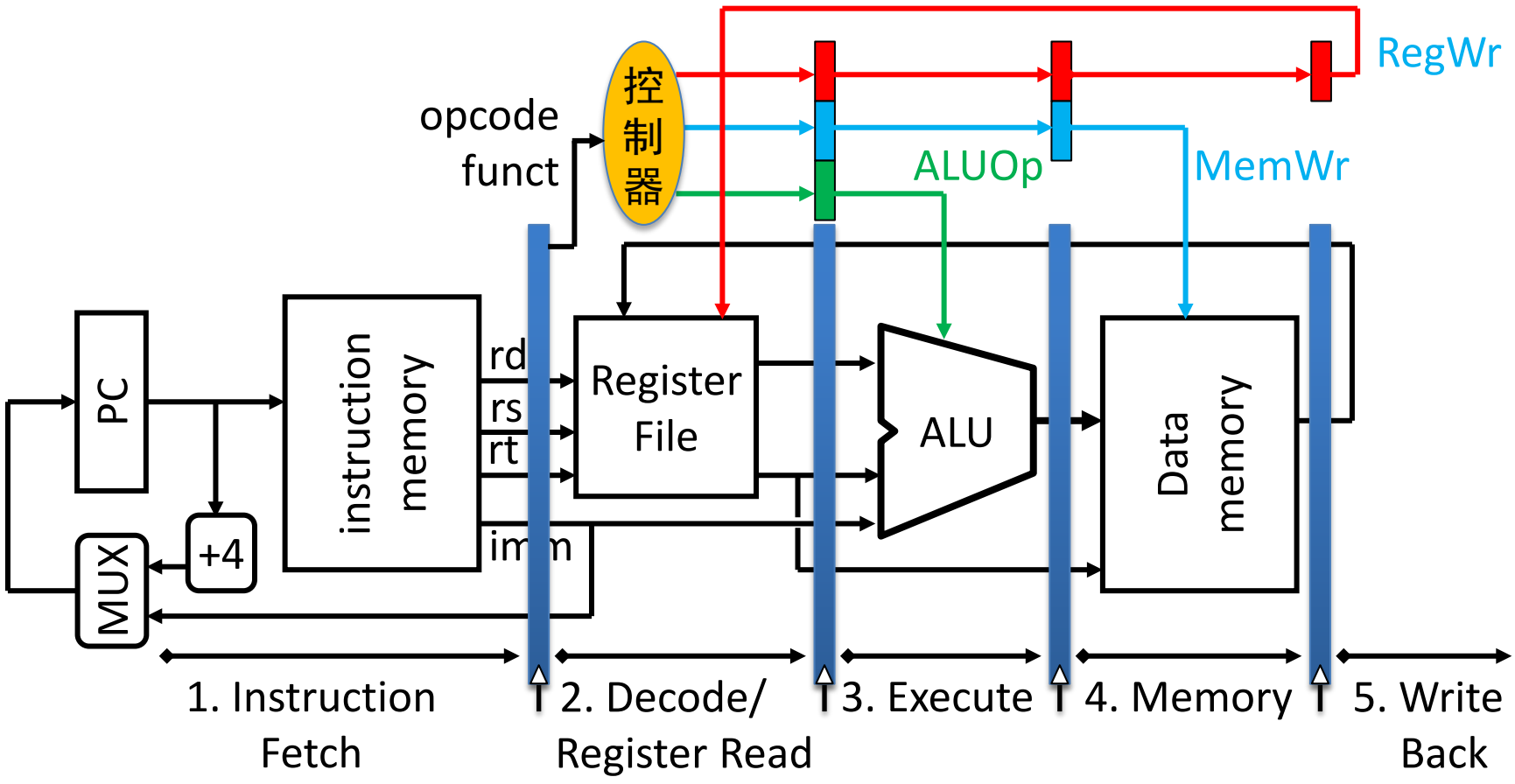
LW
SW
ADDU
SUBU
ORI
LUI
BEQ
J
JAL
JALR

提纲

- 基础指令集
- 流水线基本架构
- 无转发数据通路构造方法
- 功能部件控制信号构造方法
- 数据冒险的一般性分析方法
- 暂停机制生成方法
- 转发机制生成方法
- 控制冒险处理机制

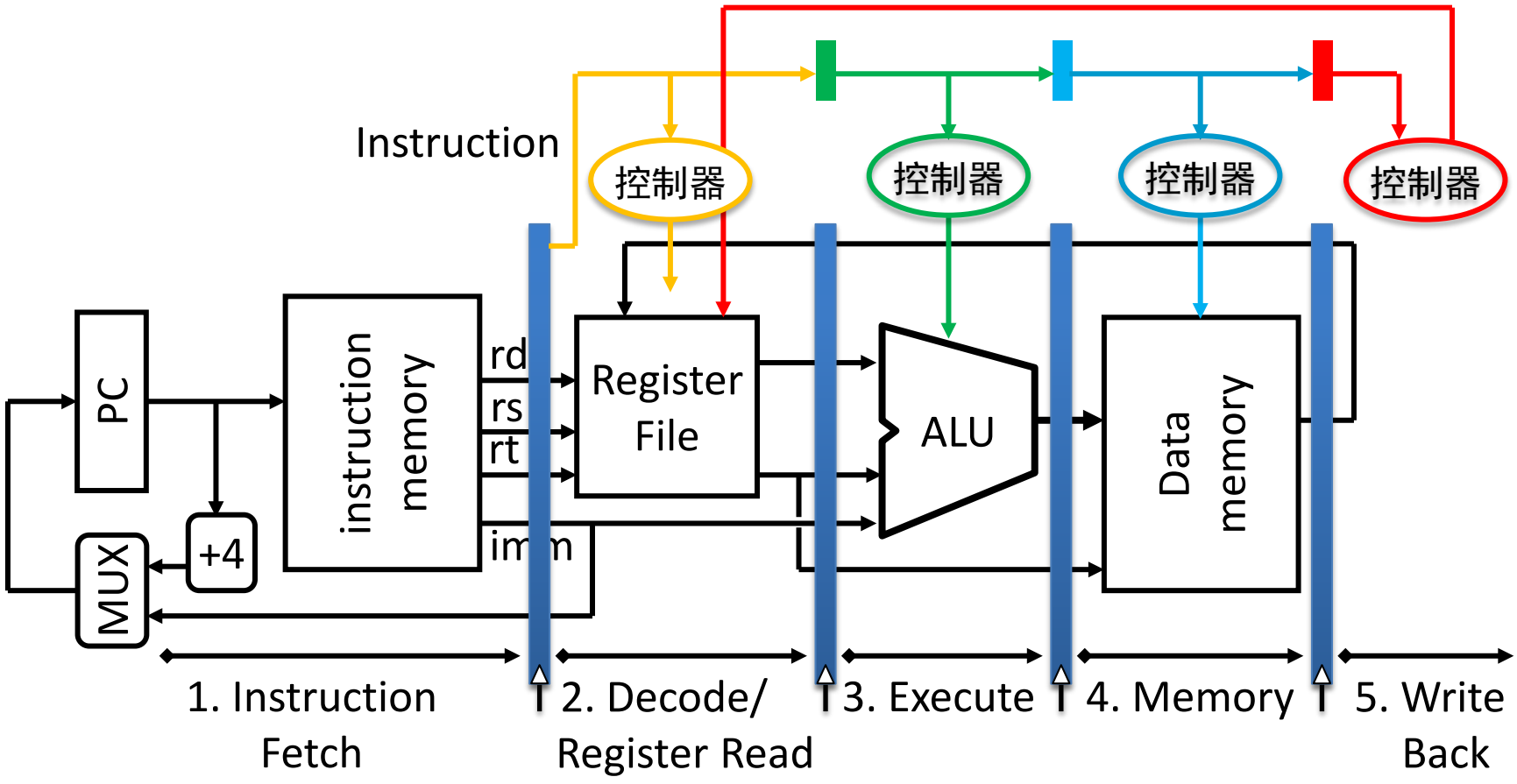
集中式控制器与分布式控制器

- 集中式控制器：流水控制信号（用信号代表指令）
  - 控制器只在ID阶段
  - 控制器产生全部的译码信号
  - 流水所有的控制信号（译码信号）



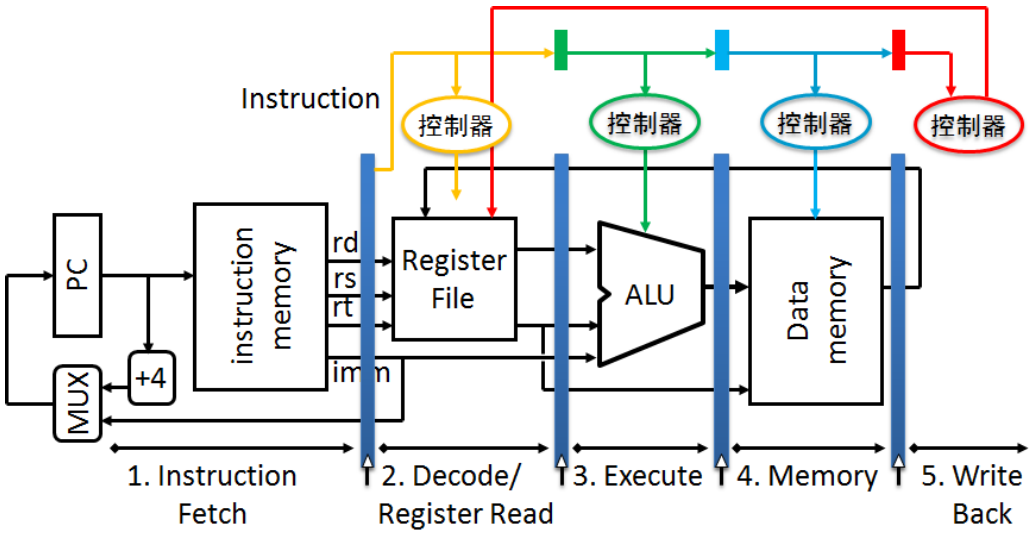
集中式控制器与分布式控制器

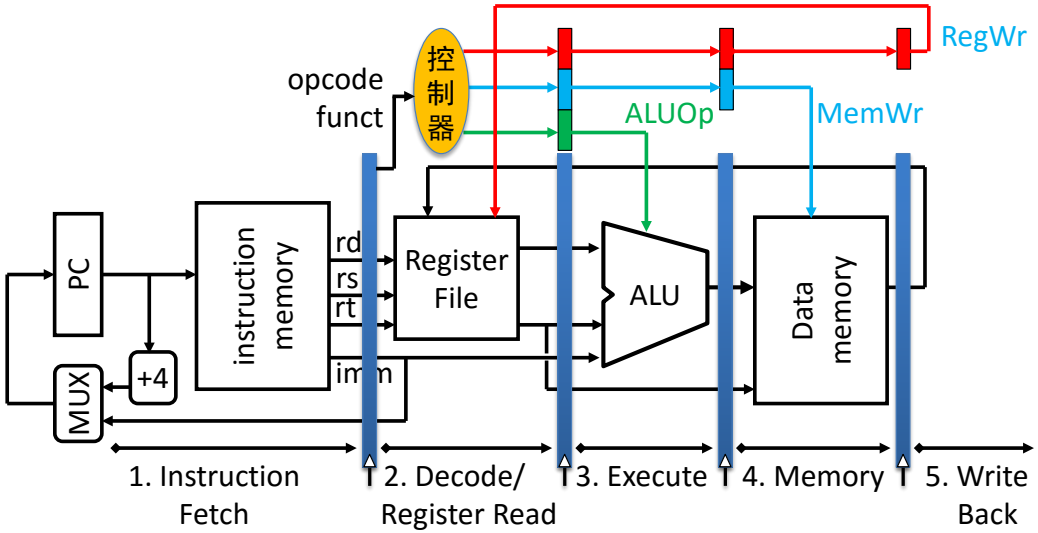
- 分布式式控制器：流水指令
  - 控制器分布在多个流水线阶段
  - 每级控制器只产生该级功能部件相关的译码信号
    - 各级控制器都是单周期控制器的子集



集中式控制器与分布式控制器

- 资源使用：集中式控制器
- 结构简洁：分布式控制器
  - “马路警察，各管一段”
- 代码理解：分布式控制器
  - 写起来字符多些，但更易于理解语义
- 项目维护：分布式控制器
  - 更有利于多人协同





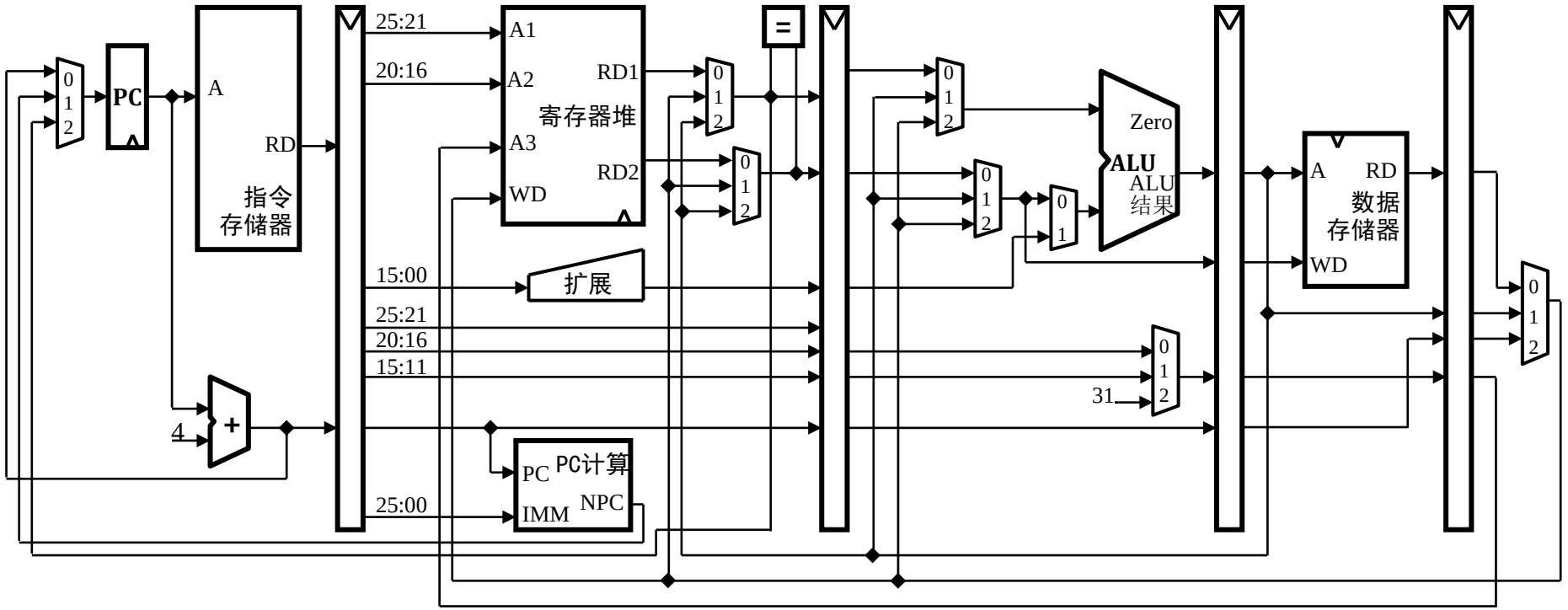
课程讲授：分布式
（更清晰的表达设计思路）
实际开发：随意选择

3控制器架构

- 功能部件控制器（就是书中的控制器）
  - 译码指令，控制各个功能部件
  - 属于功能设计范畴：与指令的功能相关，与性能无关
    - 无论单周期还是流水线，设计思路相同
- 暂停控制器
  - 将IF/ID指令与前序指令（位于后序流水段）分析，决定是否暂停
  - 属于性能设计范畴
- 转发控制器
  - 分析各级指令的相关性，决定如何转发
  - 属于性能设计范畴
- 三控制器架构特点
  - 结构清晰，易于理解，互不干扰

基础型流水线

- 以性能为目标
  - 数据冒险：转发、暂停
  - 控制冒险：分支比较前移、转发、暂停
- 功能并不完善，用于作为架构参考
  - 并不能100%正确的支持基础指令集



基础架构抽象

- 延用单周期数据通路功能部件
  - 开发过程中需要进一步增加部件及完善部件功能
- 按流水段分类，便于理解和记忆
- RF在2个阶段均被使用
  - 译码/读操作数阶段；结果回写寄存器阶段

阶段	部件	输入	输出	描述
取指令（F级）	PC	D	Q	标准D寄存器
	ADD4	PC，+4	PC4	完成PC+4
	IM	A	D	指令存储器
译码/读操作数(D级)	RF	A1，A2，A3，WD	RD1，RD2	寄存器堆
	EXT	IMM16	IMM32	立即数扩展
	NPC	PC，IMM26	NextPC	为B类/J计算下条地址
	CMP	D1，D2	Result	比较2个数
计算（E级）	ALU	A，B	ALU	算数/逻辑运算
访存（M级）	DM	A，WD	RD	数据存储器
回写（W级）	RF			

流水线寄存器

- 设置4级流水线寄存器
  - 5级流水线的最后一级寄存器为RF
- 标记X：代表对应流水级需要设置相应寄存器
  - IR：4个流水级均需要
  - AO：仅M级和W级需要

X级寄存器：是从读出角度看（例如：M级就是EX/MEM）

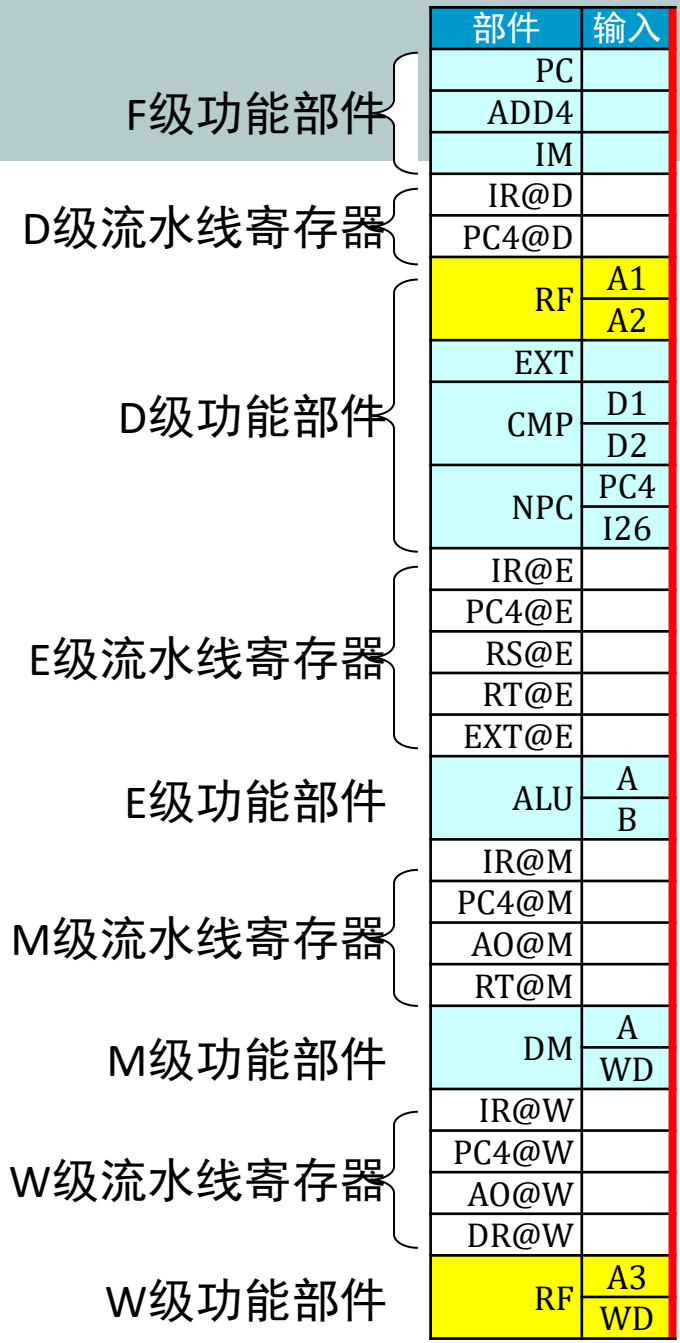
名称	功能	D级 IF/ID	E级 ID/EX	M级 EX/MEM	W级 MEM/WB
IR	传递指令	X	X	X	X
PC4	下一条指令地址	X	X	X	X
RS	RF的RS值(RD1输出)		X		
RT	RF的RT值(RD2输出)		X	X	
EXT	扩展后的32位立即数		X		
AO	ALU计算结果			X	X
DR	DM读出结果				X

提纲

- 基础指令集
- 流水线基本架构
- 无转发数据通路构造方法
- 功能部件控制信号构造方法
- 数据冒险的一般性分析方法
- 暂停机制生成方法
- 转发机制生成方法
- 控制冒险处理机制

流水线数据通路构造表格

- 每级由寄存器和功能部件组成
  - 按流水线5个阶段划分
- X@Y：代表Y阶段的X寄存器
  - IR@W：W级的IR
- RF：人为的部署在2个阶段（方便理解）
  - D阶段：准备操作数
  - W阶段：回写结果



S1：逐条指令构造数据通路

- 根据RTL描述，反推各级流水线寄存器、功能部件间连接关系
  - LW：5级
- IR必填
  - 采用分布式译码
- 指令不涉及的不需要填：如PC4
- 表示方法
  - X[y]：代表X部件的y域
  - IR@D[i16]：D级IR的16位立即数

部件	输入	LW
PC		ADD4
ADD4		PC
IM		PC
IR@D		IM
PC4@D		
RF	A1	IR@D[rs]
	A2	
EXT		IR@D[i16]
NPC	PC4	
	I26	
IR@E		IR@D
PC4@E		
RS@E		RF.RD1
RT@E		
EXT@E		EXT
ALU	A	RS@E
	B	EXT@E
IR@M		IR@E
PC4@M		
AO@M		ALU
RT@M		
DM	A	AO@M
	WD	
IR@W		IR@M
PC4@W		
AO@W		
DR@W		DM
RF	A3	IR@W[rt]
	WD	DR@W

S1：逐条指令构造数据通路

部件	输入	LW	SW	ADDU	SUBU	ORI	BEQ	J	JAL	JALR
PC		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4|NPC	ADD4|NPC	ADD4|NPC	ADD4|RF.RD1
ADD4		PC	PC	PC	PC	PC	PC	PC	PC	PC
IM		PC	PC	PC	PC	PC	PC	PC	PC	PC
IR@D		IM	IM	IM	IM	IM	IM	IM	IM	IM
PC4@D							ADD4	ADD4	ADD4	ADD4
RF	A1	IR@D[rs]	IR@D[rs]	IR@D[rs]	IR@D[rs]	IR@D[rs]	IR@D[rs]	IR@D[rs]		IR@D[rs]
	A2			IR@D[rt]	IR@D[rt]		IR@D[rt]	IR@D[rt]		
EXT		IR@D[i16]	IR@D[i16]			IR@D[i16]				
CMP	D1						RF.RD1			
	D2						RF.RD2			
NPC	PC4						PC4@D	PC4@D	PC4@D	
	I26						IR@D[i16]	IR@D[i26]	IR@D[i26]	
IR@E		IR@D	IR@D	IR@D	IR@D	IR@D			IR@D	IR@D
PC4@E									PC4@D	PC4@D
RS@E		RF.RD1	RF.RD1	RF.RD1	RF.RD1	RF.RD1				
RT@E			RF.RD2	RF.RD2	RF.RD2					
EXT@E		EXT	EXT			EXT				
ALU	A	RS@E	RS@E	RS@E	RS@E	RS@E				
	B	EXT@E	EXT@E	RT@E	RT@E	EXT@E				
IR@M		IR@E	IR@E	IR@E	IR@E	IR@E			IR@E	IR@E
PC4@M									PC4@E	PC4@E
AO@M		ALU	ALU	ALU	ALU	ALU				
RT@M			RT@E							
DM	A	AO@M	AO@M							
	WD		RT@M							
IR@W		IR@M		IR@M	IR@M	IR@M			IR@M	IR@M
PC4@W									PC4@M	PC4@M
AO@W				AO@M	AO@M	AO@M				
DR@W		DM								
RF	A3	IR@W[rt]		IR@W[rd]	IR@W[rd]	IR@W[rt]			0x1F	IR@W[rd]
	WD	DR@W		AO@W	AO@W	AO@W			PC4@W	PC4@W

S2：综合全部指令的数据通路

- 水平方向归并
  - 去除冗余输入来源
- 在每个输入来源个数大于1的输入端前增加1个MUX
  - 注意：同时需要产生相应的控制信号
- 特例：NPC的i16和i26归并为i26

部件	输入	输入来源			MUX	控制
PC		ADD4	NPC	RF.RD1	M1	PCSel
ADD4		PC				
IM		PC				
IR@D		IM				
PC4@D		ADD4				
RF	A1	IR@D[rs]				
	A2	IR@D[rt]				
EXT		IR@D[i16]				
CMP	D1	RF.RD1				
	D2	RF.RD2				
NPC	PC4	PC4@D				
	I26	IR@D[i26]				
IR@E		IR@D				
PC4@E		PC4@D				
RS@E		RF.RD1				
RT@E		RF.RD2				
EXT@E		EXT				
ALU	A	RS@E				
	B	EXT@E	RT@E		M2	BSel
IR@M		IR@E				
PC4@M		PC4@E				
AO@M		ALU				
RT@M		RT@E				
DM	A	AO@M				
	WD	RT@M				
IR@W		IR@M				
PC4@W		PC4@M				
AO@W		AO@M				
DR@W		DM				
RF	A3	IR@W[rt]	IR@W[rd]	0x1F	M3	WRSel
	WD	DR@W	AO@W	PC4@W	M4	WDSel

提纲

- 基础指令集
- 流水线基本架构
- 无转发数据通路构造方法
- 数据冒险的一般性分析方法
- 暂停机制生成方法
- 转发机制生成方法
- 控制冒险处理机制

数据冒险：需求与供给能否匹配？

- 需求者：需要引用reg值的component
  - 例如：所有运算类指令的需求在E级的ALU
  - 例如：j指令不需要读取任何GPR，因此j指令没有需求
- 供给者：保存有reg新结果的流水线寄存器
  - 例如：所有运算类指令的供给者是EX/MEM、MEM/WB
  - 例如：load类指令的供给者是MEM/WB
- 数据冒险可以转化为：需求与供给的匹配
  - 转发：结果已经保存在EX/MEM或MEM/WB
  - 暂停：结果尚未出现

- Q：如果有多个供给者，EX/MEM与MEM/WB哪个值更新？

需求者的最晚时间模型

- T_{use}(time-to-use)：指令进入IF/ID寄存器后，其后的某个功能部件再经过多少cycle就必须要使用寄存器值
  - 特点1：静态值，读取操作数的时间上限
  - 特点2：同一条指令可以有2个不同的T_{use}
  - 例如，R型计算类指令的T_{use}为1
    - rs/rt值：最晚被ID/EX寄存器驱动
  - 例如，store型计算类指令的T_{use}分别为1和2
    - rs值：最晚被ID/EX寄存器驱动
    - rt值：最晚被EX/MEM寄存器驱动

供给者的最早时间模型

- T_{new}(time-to-new)：位于ID/EX及其后各流水线的指令，再经过多少周期能够产生要写入寄存器的结果
  - 特点1：动态值，随着指令的流动，该值在不断减小，直至0
  - 特点2：一条指令可以有多个不同的T_{new}
  - 例如，R型计算类指令的T_{new}为1或0
    - 1：指令位于ID/EX，ALU正在计算
    - 0：指令位于EX/MEM和MEM/WB（结果已经产生）
  - 例如，load型计算类指令的T_{new}为2，1，0
    - 2：指令位于ID/EX，尚未读取存储器
    - 1：指令位于EX/MEM，正在读取存储器
    - 0：指令位于MEM/WB，结果已经产生

数据冒险的策略分析

- $T_{new} = 0$：表明结果已经产生
  - 指令位于MEM/WB：那么虽然结果尚未最终写入RF，但RF设计使得W结果可以被正确的读出，因此无需任何操作（实质是RF内部转发）
  - 指令位于其他位置：通过转发解决数据相关
- $T_{new} \neq 0$：表明结果尚未产生
  - $T_{new} > T_{use}$：不可能及时供给数据，只能暂停流水线
  - $T_{new} \leq T_{use}$：由于结果产生时间短于读取时间，因此当结果产生后可以通过转发解决数据冒险

- 暂停：$T_{new} > T_{use}$
- 转发：$T_{new} = 0$&指令不在MEM/WB 或 $T_{new} \leq T_{use}$

数据冒险的策略分析

- 思想：在门口检查暂停；放进去的都能执行
  - 暂停指令被放进去也不能执行，那为什么要放进去呢？！
- 思路：先解决暂停，再解决转发
  - 去除暂停部分后，有助于简化转发的分析复杂度
- 暂停：在IF/ID决定是否需要暂停
  - 只需将指令的T_{use}与各级的T_{new}进行对比即可决定是否需要暂停
  - 分析量少
- 转发：由于在D级、E级、M级均涉及操作数读取
  - 需要将各级指令与其后的各级指令进行对比
  - 分析量大

提纲

- 基础指令集
- 流水线基本架构
- 无转发数据通路构造方法
- 数据冒险的一般性分析方法
- 暂停机制生成方法
- 转发机制生成方法
- 控制冒险处理机制

构造T_{use}表和T_{new}表

ADD
SUB
andi
ori
LW
SW
BEQ

- 示例指令集
  - add，sub：cal_r类，即R型计算类指令
  - andi，ori：cal_i类，即I型计算类指令
  - beq：b_type类
  - lw：ld类
  - sw：st类
- 会产生结果的指令：cal_r类，cal_i类，load类
- 用指令分类可以大幅度简化分析工作量

```
cal_r = add  + sub + or   + …
cal_i = addi + ori + andi + …
ld    = lw   + lb  + lh   + …
st    = sw   + sb  + …
```

构造T_{use}表和T_{new}表

- T_{use}表：以指令位于IF/ID来分析
  - 流水线在指令被存储在IF/ID后就决定是否需要暂停
- T_{new}表：只需分析指令处于ID/EX和EX/MEM这2种情况
  - IF/ID：无任何结果
  - MEM/WB：如果指令（即结果）到达该阶段，则RF就可以消除数据冒险

IF/ID当前指令		
指令类型	源寄存器	Tuse
beq	rs/rt	0
cal_r	rs/rt	1
cal_i	rs	1
load	rs	1
store	rs	1
store	rt	2

ID/EX (Tnew)			EX/MEM (Tnew)			MEM/WB (Tnew)		
cal_r 1/rd	cal_i 1/rt	load 2/rt	cal_r 0/rd	cal_i 0/rt	load 1/rt	cal_r 0/rd	cal_i 0/rt	load 0/rt

2:2个cycle后产生结果
rt：结果要写入rt寄存器

构造阻塞矩阵

- 凡是$T_{new} > T_{use}$的指令序列，都需要阻塞
- 示例
  - **序列1 cal_r – beq**：由于cal_r需要1个cycle后才能得到结果，而beq现在就需要读取寄存器，因此只能暂停
  - **序列2 load – store**：store要读取的rs将在1个cycle后必须使用，而位于ID/EX的load必须经过2个cycle后才能读出DM的数据，因此只能暂停

IF/ID当前指令			ID/EX (T_{new})			EX/MEM (T_{new})
指令类型	源寄存器	T_{use}	cal_r 1/rd	cal_i 1/rt	load 2/rt	load 1/rt
beq	rs/rt	0	暂停	暂停	暂停	暂停
cal_r	rs/rt	1			暂停	
cal_i	rs	1			暂停	
load	rs	1			暂停	
store	rs	1			暂停	

构造阻塞矩阵

- 凡是$T_{new} > T_{use}$的指令序列，都需要阻塞
- 示例
  - **序列1 cal_r – beq**：由于cal_r需要1个cycle后才能得到结果，而beq现在就需要读取寄存器，因此只能暂停
  - **序列2 load – store**：store要读取的rs将在1个cycle后必须使用，而位于ID/EX的load必须经过2个cycle后才能读出DM的数据，因此只能暂停
    - 下例说明了不暂停会产生错误（lw在cycle4产生结果，sw在cycle3要用结果）

地址	指令
0	`lw $1, 0($2)`
4	`sw $3, 0($1)`
	...
	...

		IF级	ID级	EX级	MEM级	WB级	
CLK	PC	IM	IF/ID	ID/EX	EX/MEM	MEM/WB	RF
↑ 1	0→4	lw→sw	lw				
↑ 2	4→8	sw→	sw	lw			
↑ 3	...	...	...	sw用rs	lw		
↑ 4	...	...	...			lw结果	

暂停控制信号

- 建立分类指令的暂停条件

```
stall_b = IR_D.op==beq &
          (cal_r_E & ((IR_D.rs==IR_E.rd) | (IR_D.rt==IR_E.rd)) |
           ......
          (ld_E    & ((IR_D.rs==IR_M.rt) | (IR_D.rt==IR_M.rt)))
stall_cal_r = cal_i_D & load_E & ((IR_D.rs==IR_E.rt) |
                                  (IR_D.rt==IR_E.rt) )
```

- 建立最终的暂停条件

```
stall = stall_b + …
```

IF/ID当前指令			ID/EX (T_{new})			EX/MEM (T_{new})
指令类型	源寄存器	T_{use}	cal_r 1/rd	cal_i 1/rt	load 2/rt	load 1/rt
beq	rs/rt	0	暂停	暂停	暂停	暂停
cal_r	rs/rt	1			暂停	
cal_i	rs	1			暂停	
load	rs	1			暂停	
store	rs	1			暂停	

暂停控制信号

- 执行动作：
  - ①冻结IF/ID：sub继续被保存
  - ②清除ID/EX：指令全为0，等价于插入NOP
  - ③禁止PC：防止PC继续计数，PC应保持为PC+4

```
IR_D.en  = !stall
IR_E.clr =  stall
PC.en    = !stall
```

提纲

- 基础指令集
- 流水线基本架构
- 无转发数据通路构造方法
- 数据冒险的一般性分析方法
- 暂停机制生成方法
- 转发机制生成方法
- 控制冒险处理机制

转发机制生成方法

- S1：根据T_{use}和T_{new}构造转发MUX
- S2：构造每个转发MUX的控制信号表达式

根据Tuse和Tnew构造每个转发MUX

- 按照指令分类，梳理指令在各级流水线的rs或rt读需求
  - 例如：beq因在D阶段前置判断条件，故需要用到D级IR的rs/rt域
- 每个读需求点：配置1个转发MUX
- 转发MUX的输入0：必然是本级流水线寄存器
  - 对于D级转发MUX来说，本级流水线寄存器就是RF
- 忠告：命名必须遵循一定的规则
  - **MFRSD： RS寄存器值在D级的转发MUX**

流水线寄存器	源寄存器	涉及指令
IR@D	rs	beq
	rt	beq
IR@E	rs	cal_r, cal_i, ld, st
	rt	cal_r, st
IR@M	rt	st

转发MUX	控制信号	输入0
MFRSD	ForwardRSD	RF.RD1
MFRTD	ForwardRTD	RF.RD2
MFRSE	ForwardRSE	RS@E
MFRTE	ForwardRTE	RT@E
MFRTM	ForwardRTM	RT@M

根据Tuse和Tnew构造每个转发MUX

- 生成转发MUX分析矩阵：只保留Tnew=0项
  - 非0项：已经用于暂停处理了

ID/EX (Tnew)			EX/MEM (Tnew)			MEM/WB (Tnew)		
cal_r 1/rd	cal_i 1/rt	load 2/rt	cal_r 0/rd	cal_i 0/rt	load 1/rt	cal_r 0/rd	cal_i 0/rt	load 0/rt

流水线寄存器	源寄存器	涉及指令				EX/MEM (Tnew)		MEM/WB (Tnew)		
						cal_r 0/rd	cal_i 0/rt	cal_r 0/rd	cal_i 0/rt	load 0/rt
IR@D	rs	beq	MFRSD	ForwardRSD	RF.RD1					
	rt	beq	MFRTD	ForwardRTD	RF.RD2					
IR@E	rs	cal_r, cal_i, ld, st	MFRSE	ForwardRSE	RS@E					
	rt	cal_r, st	MFRTE	ForwardRTE	RT@E					
IR@M	rt	st	MFRTM	ForwardRTM	RT@M					
			转发MUX	控制信号	输入0					

根据Tuse和Tnew构造每个转发MUX

- 构造每个转发MUX的后续输入
- 示例：MFRSD
  - EX/MEM：cal_r和cal_i指令都是计算类，结果必然由ALU产生，因此均填入**AO**。即代表MFRSD的输入来自EX/MEM中的AO寄存器
    - AO：代表ALUOut
  - MEM/WB：由于这是最后一级，即所有指令的结果都通过M4(MUX)回写，因此均填入**M4**。

流水线寄存器	源寄存器	涉及指令	转发MUX	控制信号	输入0	EX/MEM (Tnew)		MEM/WB (Tnew)		
						cal_r 0/rd	cal_i 0/rt	cal_r 0/rd	cal_i 0/rt	load 0/rt
IR@D	rs	beq	MFRSD	ForwardRSD	RF.RD1	**AO**	**AO**	**M4**	**M4**	**M4**
	rt	beq	MFRTD	ForwardRTD	RF.RD2					
IR@E	rs	cal_r, cal_i, ld, st	MFRSE	ForwardRSE	RS@E					
	rt	cal_r, st	MFRTE	ForwardRTE	RT@E					
IR@M	rt	st	MFRTM	ForwardRTM	RT@M					

根据Tuse和Tnew构造每个转发MUX

- 根据前例，可以构造出全部的转发MUX
  - 当store类指令位于EX/MEM时，该级不可能再有其他指令了，故有2项空白
- 构造更大指令集时，需求项及供给项可能均需要调整
  - 但由于MIPS的指令功能到格式映射的相对统一，因此调整不会剧烈
  - 再次从一个侧面反映出MIPS指令集设计的水平！

流水线寄存器	源寄存器	涉及指令				EX/MEM (Tnew)		MEM/WB (Tnew)		
						cal_r 0/rd	cal_i 0/rt	cal_r 0/rd	cal_i 0/rt	ld 0/rt
IR@D	rs	beq	MFRSD	ForwardRSD	RF.RD1	AO	AO	M4	M4	M4
	rt	beq	MFRTD	ForwardRTD	RF.RD2	AO	AO	M4	M4	M4
IR@E	rs	cal_r, cal_i, ld, st	MFRSE	ForwardRSE	RS@E	AO	AO	M4	M4	M4
	rt	cal_r, st	MFRTE	ForwardRTE	RT@E	AO	AO	M4	M4	M4
IR@M	rt	st	MFRTM	ForwardRTM	RT@M			M4	M4	M4
			转发MUX	控制信号	输入0					

根据Tuse和Tnew构造每个转发MUX

- 对于MFRSD来说，其最终有效输入为3个
  - 输入0～RF.RD1；输入1～AO；输入2～M4
- 实现转发MUX时，需要剔除每级中的重复项
- 在表格中保留重复项的目的在于有利于建立后续的控制信号方程

输入	来源
0	RF.RD1
1	AO@M
2	M4@W

MFRSD	ForwardRSD	RF.RD1	AO	AO	M4	M4	M4
MFRTD	ForwardRTD	RF.RD2	AO	AO	M4	M4	M4
MFRSE	ForwardRSE	RS@E	AO	AO	M4	M4	M4
MFRTE	ForwardRTE	RT@E	AO	AO	M4	M4	M4
MFRTM	ForwardRTM	RT@M			M4	M4	M4
转发MUX	控制信号	输入0					

MFRSD	ForwardRSD	RF.RD1	AO@M	M4
MFRTD	ForwardRTD	RF.RD2	AO@M	M4
MFRSE	ForwardRSE	RS@E	AO@M	M4
MFRTE	ForwardRTE	RT@E	AO@M	M4
MFRTM	ForwardRTM	RT@M	M4	
转发MUX	控制信号	输入0	输入1	输入2

数据通路增加转发MUX

- 遍历数据通路的功能部件，找到所有出现rs和rt的需求点
- 注意：ALU.B和RT@M，这两个rt需求是相同的！
  - 这意味着它们应该来自同一个转发MUX
- 注意：对于PC，由于本小节示例指令集中没有j/jal/jalr指令，因此无相应的转发MUX

部件	输入	输入来源		MUX	控制
PC		ADD4	NPC	M1	PCSel
ADD4		PC			
IM		PC			
IR@D		IM			
PC4@D		ADD4			
RF	A1	IR@D[rs]			
	A2	IR@D[rt]			
EXT		IR@D[i16]			
CMP	D1	RF.RD1			
	D2	RF.RD2			
NPC	PC4	PC4@D			
	I26	IR@D[i26]			
IR@E		IR@D			
PC4@E		PC4@D			
RS@E		RF.RD1			
RT@E		RF.RD2			
EXT@E		EXT			
ALU	A	RS@E			
	B	EXT@E	RT@E	M2	BSel
IR@M		IR@E			
PC4@M		PC4@E			
AO@M		ALU			
RT@M		RT@E			
DM	A	AO@M			
	WD	RT@M			
IR@W		IR@M			
PC4@W		PC4@M			
AO@W		AO@M			
DR@W		DM			
RF	A3	IR@W[rt]	IR@W[rd]	M3	WRSel
	WD	DR@W	AO@W	M4	WDSel

数据通路增加转发MUX

- 遍历数据通路的功能部件，找到所有出现rs和rt的需求点
- 将对应的输入替换为转发MUX的输出
  - 注意ALU.B和RT@M，这两个rt需求是相同的，因此应该用同一个转发MUX

部件	输入	输入来源		MUX	控制
PC		ADD4	NPC	M1	PCSel
ADD4		PC			
IM		PC			
IR@D		IM			
PC4@D		ADD4			
RF	A1	IR@D[rs]			
	A2	IR@D[rt]			
EXT		IR@D[i16]			
CMP	D1	MFRSD			
	D2	MFRTD			
NPC	PC4	PC4@D			
	I26	IR@D[i26]			
IR@E		IR@D			
PC4@E		PC4@D			
RS@E		RF.RD1			
RT@E		RF.RD2			
EXT@E		EXT			
ALU	A	MFRSE			
	B	EXT@E	MFRTE	M2	BSel
IR@M		IR@E			
PC4@M		PC4@E			
AO@M		ALU			
RT@M		MFRTE			
DM	A	AO@M			
	WD	MFRTM			
IR@W		IR@M			
PC4@W		PC4@M			
AO@W		AO@M			
DR@W		DM			
RF	A3	IR@W[rt]	IR@W[rd]	M3	WRSel
	WD	DR@W	AO@W	M4	WDSel

MFRSD	RF.RD1	AO@M	M4
MFRTD	RF.RD2	AO@M	M4
MFRSE	RS@E	AO@M	M4
MFRTE	RT@E	AO@M	M4
MFRTM	RT@M	M4	
转发MUX	输入0	输入1	输入2

转发机制生成方法

- S1：根据Tuse和Tnew构造每个转发MUX
- S2：构造每个转发MUX的控制信号表达式

S2：构造每个转发MUX的控制信号表达式

- 控制信号表达式构造的基本思路
  - 精确控制每个转发选择
  - 缺省选择输入0

输入	来源
0	RF.RD1
1	AO@M
2	M4@W

- 注意：M阶段的rt

						EX/MEM (Tnew)		MEM/WB (Tnew)		
流水线寄存器	源寄存器	涉及指令				cal_r 0/rd	cal_i 0/rt	cal_r 0/rd	cal_i 0/rt	ld 0/rt
IR@D	rs	beq	MFRSD	ForwardRSD	RF.RD1	AO	AO	M4	M4	M4
	rt	beq	MFRTD	ForwardRTD	RF.RD2	AO	AO	M4	M4	M4
IR@E	rs	cal_r, cal_i, ld, st	MFRSE	ForwardRSE	ID/EX.RS	AO	AO	M4	M4	M4
	rt	cal_r, st	MFRTE	ForwardRTE	ID/EX.RT	AO	AO	M4	M4	M4
IR@M	rt	st	MFRTM	ForwardRTM	EX/MEM.RT			M4	M4	M4
			转发MUX	控制信号	输入0					

示例：always语句建模MF_RS_D的控制信号表达式

- 顺序代表优先级
  - 当多条前序指令写同一个寄存器，距离D级越近优先级越高

输入	来源
0	RF.RD1
1	AO@M
2	M4@W

Hign

优先级

Low

```
assign ForwardRSD[1:0] =
    IR_D.op==beq & cal_r_M & IR_D[`rs]==IR_M[`rd] ? 1 :
    IR_D.op==beq & cal_i_M & IR_D[`rs]==IR_M[`rt] ? 1 :
    IR_D.op==beq & cal_r_W & IR_D[`rs]==IR_W[`rd] ? 2 :
    IR_D.op==beq & cal_i_W & IR_D[`rs]==IR_W[`rt] ? 2 :
    IR_D.op==beq & load_W  & IR_D[`rs]==IR_W[`rt] ? 2 :
                                                    0 ;
```

流水线寄存器	源寄存器	涉及指令				EX/MEM (Tnew)		MEM/WB (Tnew)		
						cal_r 0/rd	cal_i 0/rt	cal_r 0/rd	cal_i 0/rt	ld 0/rt
IF/ID	rs	beq	MFRSD	ForwardRSD	RF.RD1	AO	AO	M4	M4	M4
	rt	beq	MFRTD	ForwardRTD	RF.RD2	AO	AO	M4	M4	M4
			转发MUX	控制信号	输入0					

提纲

- 基础指令集
- 流水线基本架构
- 无转发数据通路构造方法
- 数据冒险的一般性分析方法
- 暂停机制生成方法
- 转发机制生成方法
- 控制冒险处理机制

控制冒险处理机制

- 分歧点1：是否实现延迟槽
  - 如果实现，需要注意jal及jalr指令应保存PC+8(或者更多，取决于是否前移)
- 分歧点2：执行是否前移至ID阶段
- 课程要求：实现延迟槽，并且前移至ID阶段

延迟槽 / 前移	是	否
是	硬件无需处理 编译调度指令	B类：有条件清除IF/ID J类：无条件清除IF/ID
否		B类：有条件清除IF/ID、ID/EX J类：无条件清除IF/ID、ID/EX、EX/MEM

Q：JAL、JALR的回写寄存器怎么处理呢？

A：视同普通的回写

总结

- 流水线设计的复杂性在于对冲突的覆盖性分析
  - 覆盖性分析使得设计与测试均具备了完整的正向设计的理论基础
  - 缺乏覆盖性分析，就不能断言是否处理所有冲突
  - 避免了频繁的、无谓的试错
  - 提高开发效率，确保开发正确性
- 教科书的不足
  - 没有覆盖性分析，难以满足大规模指令集的流水线设计与测试需求
  - 没有覆盖性分析，必然遗漏部分数据相关
    - 如lw～sw指令，必须暂停。但事实上可以通过增加转发MUX实现不停顿
    - 如cal～sw指令，未明确指出处理机制
  - RF内部的数据转发语焉不详
    - 内部转发：当读和写同一个寄存器时，读出的数据应该为要写入的数据

忠告

- 设计越细致、越充分越好！
- 所有的设计规划都会有回报的（是加速型回报）！
- 不要急于编码！

- 设计越细致、越充分越好！不要急于编码！
- 所有的设计规划都会有回报的（是加速型回报）！
- 不要急于编码！

- 设计越细致、越充分越好！不要急于编码！
- 所有的设计规划都会有回报的（是加速型回报）！
- 不要急于编码！